SW	HIR	147	C

HIRDLS

HIGH RESOLUTION DYNAMICS LIMB SOUNDER

Originators: J.J. Barnett, A. G. Darbyshire, R.J.Wells Date: 01 DEC 21

Subject/title: SAIL REQUIREMENTS DOCUMENT

Description/summary/contents:

Requirements document for Science Algorithm Implementation Language (SAIL)

Key words: Control, Language, Software, Requirements

Purpose of this document: Project Use

Reviewed/approved by: Date (yy MMM dd):			

OXFORD UNIVERSITY Atmospheric, Oceanic and Planetary Physics Department of Physics Parks Road OXFORD OX1 3PU United Kingdom.	LOCKHEED MARTIN MISSILES & SPACE Advanced Technology Center 3251 Hanover Street Palo Alto, California 94304-1191 United States of America	EOS

CHANGE SHEET

23-APR-1998
Revision B Incorporate changes requested at the HIRDLS Flight Software Meeting, Oxford, March 30th - April 3rd, 1998. (MV-LOC-309).

a) Delete requirement 4.1.2 (SAIL control of TBD functions)
b) 4.1.3 change to "write to housekeeping'"
c) 4.2.11.2 change "overflow" to ~~"divide by zero"~~"error" change definition of modulo operator
d) 4.3.3 change "engineering" to "housekeeping"

26-MAY-1999
Revision B Clarify requirement for the background task (Task 15)

a) Add 4.2.15
b) 4.3.2 Change "up to 15" to "all". Delete 2nd sentence.
c) 4.3.3 Delete "non-background" and "and any background".

16 OCT 2001
Revision C Update Requirements for Acceptance Testing

a) 4.1.8 Delete requirement
b) 4.2.3 Add sentence that ground software can be written in C++
c) 4.2.15 Change "miliseconds" to chopper rotations
d) 4.3.4 Delete "(TBD kilobytes)". Size is not predetermined.
e) Add exception for HIR_SAI_SETPARAM command.
f) 4.4.3 Change "startup" to "creation"
g) 4.5.4 Delete Requirement. Implemented in IPU software.
h) 5.0 Update Requirements Traceability Matrix to reflect changes.
i) Added SAIL requirements Ids to facilitate tracking to the SAIL SW trace VCRM.
j) Added SQE to signature page.

SW-HIR-147

Science Algorithm Implementation Language (SAIL) Requirements Document

Approved by:

/s/ John Whitney, HIRDLS Program System Engineer Date

/s/ Raymond von Savoye, HIRDLS Instrument System Engineer Date

/s/ John Barnett, HIRDLS Principle Investigator Date

/s/ Jerry Drake, Flight Software Manager Date

/s/ Jim McGill, Software Quality Engineer Date

Science Algorithm Implementation Language (SAIL) Requirements Document

1. Introduction.

1.1 Purpose and Objective.

The purpose of this document is to define the requirements for that portion of the flight software resident in the Instrument Processor Subsystem (IPS) in the High Resolution Dynamics Limb Sounder (HIRDLS) experiment which will allow the instrument to be flexibly controlled through a high order language for the purpose of routine scientific observation and for the possibility of adapting the science operations on orbit to degradation in the instrument or to changing scientific observational demands. This language will be used for instrument control in four phases of the project:

a) Instrument Integration and Test (I&T)
b) Pre-launch calibration
c) Spacecraft Integration and Test (I&T)
d) Post-launch

The objective of this document is to provide sufficient guidance that when coupled with the information contained in the Command and Telemetry Handbook, the subsystem ICDs, and the SAIL documentation set, a flexible, safe and reliable method of control for the HIRDLS experiment may be produced.

2. Related Documentation.

Instrument Technical Specification (ITS) : SP-HIR-13N
Command and Telemetry Document : SP-HIR-103
Instrument Requirements Document : SC-HIR-18K
Science Requirements Document : SC-HIR-12A

3. Description of System.

Primary control of the HIRDLS instrument is through the Instrument Processor Subsystem (IPS). The Instrument Processor Unit (IPU) provides the command and telemetry interface between the instrument and the spacecraft, prepares the telemetry data, provides control over the remainder of the HIRDLS subsystems (Telescope Subsystem [TSS], Gyro Subsystem [GSS], Power Subsystem [PSS], Cooler Subsystem [CSS], In-Flight Calibration Subsystem [IFC], Operational Heater Control and Sunshield Door Control) and provides the initialization of the system on boot-up. The Signal Processing Unit [SPU], performs the Detector Control and Data Processing function.

To permit flexible, safe and reliable control of instrument functions, a software system (SAIL) is implemented in the IPU which permits the instrument to be controlled by high level language commands generated remotely.

3.1 Instrument control philosophy.

On-orbit scientific operations of the experiment requires maximal flexibility

in commanding the instrument on the one hand but also requires a highly reliable (safe) method of control which is not overly complex or time consuming to utilize. Control of scientific instruments has evolved from those with all functionality hardwired and selectable through a relatively small number of commands. This approach is highly reliable but very limited. If an implementation error is discovered after launch, a failure or degradation occurs or the objectives of the scientific observations change due to newly discovered knowledge in the field, very little or nothing can be done to adapt the instrument to the changed circumstances. Examples of such instruments are the Lockheed Hot Plasma Composition Explorer (HPCE) instrument on Dynamics Explorer 1 and the Active Magnetospheric Particle Tracers Experiment (AMPTE) on the Charge Composition Explorer (CCE) spacecraft.

The next stage of evolution was to table driven operations where many parameters related to the observation could be set through uploaded binary tables. An example of this approach is the Toroidal Imaging Mass-Angle Spectrograph (TIMAS) instrument on the Polar spacecraft of the Global Geospace Science (GGS) Mission. In this instrument, primary control is through an Instrument Control Parameter (ICP) table prescribing the priority of 23 data products for each of 32 spins of 6 seconds each. The data products themselves also contain several internal control parameters such as the number of spins over which to sum. This approach is more flexible but still limited to the choices available. It is also more complex making the generation and checking of the binary loads moderately difficult.

The next extension would be to allow as complete and flexible control of the instrument as possible through a high level language (to decrease the degree of difficulty to implement new control sequences) allowing conditional control and looping with access to the telemetry stream for decision making in the control expressions. If all commands except those which are hazardous to the instrument or those which are time critical to the fundamental operation of the instrument were allowed, this would be the fullest control extension possible. This approach was successfully implemented on the ISAMS instrument, where the high level PCL language was used to control the instrument throughout ground testing, and in orbit. The flexibility of control permitted by the PCL was essential to permit modifying instrument operation, to carry out a range of normal scientific observations, and to cope with unexpected instrument and spacecraft behaviour.

The same approach is to be used for HIRDLS. Requirement details for the language follow in this document. The command sequences generated in this language must be verified on the ground prior to upload by the responsible mission operations personnel at that time. This will ensure that the sequence(s) perform the operations as expected and that there are no unexpected error conditions or states which this sequence will elicit from the instrument. This verification is done by use of a software simulator. A Flight Software Simulator (FSS) will be produced by LMPARL and be provided to Oxford University who will add further modules for science verification. This augmented FSS will be used to verify the command sequences prior to upload. The augmented FSS itself will be verified against both the HIRDLS Engineering Model and against the Flight Model through a series of appropriately defined tests (command sequences).

4. SAIL Requirements

The Science Algorithm Implementation Language permits the flexible control of the instrument from a simple high-level language in a fashion which was proven by the ISAMS experiment to be safe and highly flexible.

The high-level language is processed on the ground into a sequence of SAIL interpreter instructions, which are uploaded to the instrument and executed under the control of the SAIL executive. The executive itself receives commands from the spacecraft; these commands can be generated either on the ground or from the spacecraft on-board computer as appropriate.

This section presents top-level requirements for the SAIL, in other words, the requirements on the functionality which the SAIL system must present to the user. The details of the implementation of the SAIL which will meet these requirements are given in the SAIL specification document.

A source is indicated for each requirement. ITS refers to the Instrument Technical Specification (SP-HIR-13N); EJ refers to Engineering Judgement, derived in large part from experience with the ISAMS PCL system and discussion meetings held in Oxford in January and September 1996.

4.1 SAIL functionality.

4.1.1 Instructions must be available in the SAIL able to control all mechanisms and issue any instrument commands, except for control of time-critical functions {SAIL010}. However, protection must be provided for a subset of commands to prevent their being issued from SAIL. The potential to override the protection by ground command must also be provided {SAIL020}.
[ITS 4.7.3.2.12.3]

4.1.2 [Deleted]

4.1.3 The SAIL must be able to write to the housekeeping data stream {SAIL050}.
[ITS 4.7.3.2.12.3]

4.1.4 There must be the ability to write to any diagnostic ports provided in the IPU hardware system {SAIL060}.
[ITS 4.7.3.2.12.3]

4.1.5 The SAIL must be able to read from all data buffers {SAIL070}.
[ITS 4.7.3.2.12.2]

4.1.6 The SAIL must have access to all relevant telemetry sampling timing information {SAIL080}, as well as access to the real time updates received from the spacecraft interface {SAIL090}.

[ITS 4.7.3.2.12.3, ITS 4.7.3.2.12.5]
4.1.7 The SAIL must have mnemonic access to all telemetry input and commands to the IPS including equator crossing/oblateness commands {SAIL100}.
[ITS 4.7.3.2.12.3]

4.1.8 ~~The SAIL must have mnemonic access to the status of the command protection system described in 4.1.1 above.~~
~~[ITS 4.7.3.2.12.3]~~Deleted

4.2 SAIL language.
It is essential that the SAIL presents to users a simple, high-level language type interface. The following requirements apply to this language interface.

4.2.1 The language must not be more complex than the minimum needed to meet the requirements described in this document. The preference is to write extra lines of code rather than have an overly complex language with consequent difficulty in testing {SAIL120}.
[ITS 4.7.1.1]

4.2.2 The SAIL language must permit extensive commenting of the source code {SAIL130}.
[EJ]

4.2.3 The SAIL language assembler and compiler must be written in ANSI standard C (or Fortran). (Non-flight code may be written in C++ per Deviation D016, GSFC 424-12-21-033.)
[ITS 3.10.1]

4.2.4 The SAIL language compiler must not employ any optimization techniques {SAIL150}.
[EJ]

4.2.5 All telemetry mnemonics in the compiler/assembler must be redefinable through a symbol table {SAIL160}.
[ITS 4.7.4]

4.2.6 The SAIL must be a strongly typed language; all variables, functions and function arguments must be declared before use {SAIL170}.
[EJ]

4.2.7 SAIL must not permit dynamic memory allocation. {SAIL180}
[EJ]

4.2.8 SAIL language statements must not permit multiple assignments in a single statement {SAIL190}.
[EJ]

4.2.9 Data types.

The data types in 4.2.9.1 to 4.2.9.6 inclusive must be provided.

4.2.9.1 Byte: unsigned 8 bits [0,255] {SAIL200}.
[ITS 4.7.3.2.12.3, EJ]

4.2.9.2 Unsigned: unsigned 16 bits [0,65535] {SAIL210}.
[ITS 4.7.3.2.12.3, EJ]

4.2.9.3 Integer: signed 32 bits [-2147483648,2147483647] {SAIL220}.
[ITS 4.7.3.2.12.3, EJ]

4.2.9.4 Float: 64 bits {SAIL230}.
[ITS 4.7.3.2.12.3, EJ]

4.2.9.5 Logical: {SAIL240}.
[ITS 4.7.3.2.12.3, EJ]

4.2.9.6 Arrays - up to 2 dimensional, of any type, with a total allocation of up to 65536 bytes SAIL250}.
[ITS 4.7.3.2.12.3, EJ]

4.2.9.7 Multiple types - where appropriate, certain areas of telemetry buffers and elsewhere must be accessible either as individual elements with associated mnemonics, or as elements of arrays of appropriate types (often 2 byte integer for 16 bit measurements). This must include the control variables used for ground communication {SAIL260}.
[ITS 4.7.3.2.12.3,EJ]

4.2.9.8. User access to pointers is prohibited {SAIL270}.
[EJ]

4.2.10 Control structures.

The following control structures must be provided:

4.2.10.1 Conditionals. A single IF-ELSE structure must be provided. Conditional structures must be able to be nested {SAIL280}.
[EJ]

4.2.10.2 Loops. A single looping structure, with appropriate loop counter testing, must be provided. Loops must be able to be nested {SAIL290}.
[EJ]

4.2.10.3 Jump. A jump to label command must be provided.
The compiler must not allow the target of a jump command to be
a) outside the same subprogram module
b) in an unrelated SAIL loop structure (4.2.10.2)
c) in a lower level SAIL loop structure nested within a SAIL loop structure
d) in a lower level conditional structure (4.2.10.1)
e) in another section of a current or higher level conditional structure {SAIL300}
[EJ]

4.2.10.4 Subprograms. The SAIL must implement subprograms {SAIL310}. Arguments to subroutines may be passed by reference or value, except in the case of arrays which must be passed by reference only. The subprogram may, but is not required to, return a value of a valid SAIL data-type [see section 4.2.9]. The language must allow the compiler to check parameter conformance between subprogram calls and declarations {SAIL320}. Recursion must not be allowed. i.e. a subprogram must not be able to call itself directly or indirectly {SAIL330}.
[EJ]

4.2.10.5 A command must be implemented to suspend execution of a task until the next nth clock period {SAIL340}. See 4.3.3
[ITS 4.7.3.2.12.4]

4.2.11 Operators

The following operators must be provided for all appropriate variable types. Using operators on inappropriate variable types must result in a compiler error.

4.2.11.1 Assignment (all types) {SAIL350}.
[EJ]

4.2.11.2 Arithmetic: add, subtract, multiply, divide, modulo (real, integer, unsigned and byte data types).
The result of i/j where i and j are integers and j is zero must be flagged as an error.
The result of a/b where a and b are reals and b is zero must be flagged as an error.
The modulo operator (%) is such that
a%b = (ABS(a) - ABS(b)*INT(a/b))*SIGN(a)*SIGN(b)
where ABS(x) is the absolute value of x
INT(x) is the largest integer less than or equal to x
SIGN(x) is -1 is x is negative, otherwise +1.
The result of a%b when b is zero must be flagged as an error.
{SAIL360}
[ITS 4.7.3.2.12.3, EJ]

4.2.11.3 Bit manipulation: shift, and, inclusive or, exclusive or (integer,unsigned and byte types) {SAIL370}.
[EJ]

4.2.11.4 Logical combination: and, inclusive or, not (logical types only) {SAIL380}.
[EJ]

4.2.11.5 Relational operators:
equality compare, inequality compare (all data types except array) less than, greater than, less than or equal to, greater than or equal to (all data types except logical and array). Result of any of these operations is a logical.
{SAIL390}
[EJ]

4.2.12 Maths functions

The following library functions must be supported: sine, cosine, exponential, square root, natural logarithm, inverse sine, inverse cosine, inverse tangent (double argument). These functions must be provided for real data types only. Also an absolute value function for both real and integer types is required.
{SAIL400}
[EJ]

4.2.13 Expressions.

Expressions in SAIL must obey the following requirements:

4.2.13.1 Expressions must be evaluated according to a predefined operator precedence {SAIL410}.
[EJ]

4.2.13.2 Evaluation of expressions involving operators of equal precedence must follow a predefined set of rules {SAIL420}.
[EJ]

4.2.13.3 Brackets must be provided to permit overriding of the predefined operator precedence {SAIL430}.
[EJ]

4.2.13.4 Mixed type expressions must be evaluated according to a predefined set of rules {SAIL440}.
[EJ]

4.2.13.5 Expressions which result in an underflow must produce a zero result, and should not be flagged as an error {SAIL450}.
[EJ]

4.2.13.6 Expressions which result in a divide-by-zero or an overflow in operation must be flagged as an error {SAIL460}.
[EJ]

4.2.14 The SAIL must be fully documented in a stand-alone document (SAIL High Level Specification Document) such that fully operational SAIL programs can be written based only on the information within that document {SAIL465}.
[EJ]

4.2.15 A SAIL task must be able to ascertain the time in ~~milliseconds~~chopper rotations
remaining before the end of the current SAIL clock period (approximately 1 second) {SAIL470}.
[EJ]

4.3 SAIL executive and task control.

The SAIL executive shall implement a set of tasks, up to 16 of which may be active at one time, and each of which must consist of a sequence of instructions. By appropriate use of task suspension instructions, all the active tasks may be serviced within a single clock period (approximately 1 second). The executive must obey the following requirements:

4.3.1 The SAIL must be able to accommodate up to 16 tasks {SAIL480}.
[EJ]

4.3.2 The SAIL must be able to complete servicing of all tasks every clock period (approximately 1 second) in a repeatable time-ordered way {SAIL490}.
[ITS 4.7.3.2.12.4,EJ]

4.3.3 To achieve quasi-parallel execution, each task is responsible for suspension of its operation each clock period. At the end of a clock period if the servicing of any active task remains uncompleted then the active task and all unserviced tasks must be suspended and the event must be flagged in the housekeeping data stream. Fully-serviced tasks should continue to run {SAIL510}.
[EJ]

4.3.4 The SAIL must dedicate a ~~(TBD Kbyte)~~ region of memory for communication between tasks and for two-way communication of parameters with external environments such as ground command and data upload. All locations in this dedicated region must have write access from the ground and must be capable of inclusion in the telemetry stream.
Each of the 16 SAIL tasks must be allocated a section of memory which it alone may write but which may be read by all tasks. These sections will need to be included in the telemetry stream at all times.
A further section must be allocated for read and write access by all tasks.
The remainder of the region must have read-only access by all tasks~~.~~, except for the HIR_SAI_SETPARAM command which any task shall be allowed to send to any other task. {SAIL520} [ITS 4.7.3.2.12.3]

4.3.5 The operation and control of the executive must be documented as part of the documentation describing how to upload and install a SAIL task {SAIL524}.
[EJ]

4.4 SAIL memory requirements.

4.4.1 Each task may be restricted to a (compiled) program memory space of 65536 bytes {SAIL540}.
[EJ]

4.4.2 Each task may be restricted to a (compiled) variable memory space of 65536 bytes {SAIL550}.
[EJ]

4.4.3 Memory allocations must be made at task ~~start-up~~creation only. Running tasks must not be able to change memory allocations {SAIL560}.
[EJ]

4.4.4 Memory protection between tasks is required, that is sections of memory must be reserved for write access exclusively by each task {SAIL570}.
[EJ]

4.4.5 The SAIL must have the capability to store spare (non-active) tasks pending future use {SAIL580}.
[EJ]

4.4.6. A function must be provided to allow SAIL to read any word of the IPU memory {SAIL590}. (This requirement does not include memory-mapped devices.)
[EJ MV-LOC-144]

4.5 SAIL command and data interface.

4.5.1 SAIL must provide sufficient information (task size etc) for memory management functions to be performed on the Instrument Support Terminal (IST). Such information should be embedded in the "micro-load" file to be up-linked to the IPU {SAIL595}.
[ITS 3.10.2.5, ITS 4.7.3.2.13, ITS 4.7.3.2.14]

4.5.2 Ground commands must be provided to permit the starting and stopping of tasks. The task start procedure will need to specify the task identifier

and necessary memory locations (code and data). The task stop procedure will specify the task identifier only. Facility must also be provided to resume a task without re-initializing the data area {SAIL600}.
[ITS 4.7.3.2.12.3, EJ]

4.5.3 The only means of starting a task must be by ground command. A task may be suspended or killed by ground command. A task may suspend, but not kill, any other task {SAIL610}. A task may also kill itself {SAIL620}.
[ITS 4.7.3.2.12.3, EJ]

4.5.4 ~~The SAIL must use a checksum or similar procedure for loading of new data tables or tasks.~~
~~[3.10.2.5, ITS 4.7.3.2.13]~~Deleted

4.5.5 The SAIL system must distinguish between, and permit the use of, data files for upload which contain complete tasks, and those which contain data tables only for use by existing tasks {SAIL640}.
[EJ]

5. Requirements Traceability Matrix

SAIL #	Short description	Origin
4.1.1	SAIL controls mechanisms	ITS 4.7.3.2.12.3
4.1.1	Provide protection and override	ITS 4.7.3.2.12.3
4.1.3	SAIL writes to housekeeping data stream	ITS 4.7.3.2.12.3
4.1.4	SAIL writes to diagnostic ports	ITS 4.7.3.2.12.3
4.1.5	SAIL reads all data buffers	ITS 4.7.3.2.12.2
4.1.6	SAIL accesses telemetry timing information	ITS 4.7.3.2.12.3
4.1.6	SAIL accesses spacecraft real-time commands	ITS 4.7.3.2.12.5
4.1.7	Mnemonic access to all input to IPS	ITS 4.7.3.2.12.3
~~4.1.8~~	~~Mnemonic access to command protections~~	~~ITS 4.7.3.2.13.3~~
4.2.1	Simple language	ITS 4.7.1.1
4.2.2	Extensive comments	EJ
4.2.3	SAIL Compiler and assembler language	ITS 3.10.1
4.2.4	Non-optimizing compiler	EJ
4.2.5	Telemetry mnemonics in redefinable table	ITS 4.7.4
4.2.6	All variables declared before use	EJ
4.2.7	No dynamic memory allocation	EJ
4.2.8	No multiple assignments	EJ
4.2.9.1	Byte data type	ITS 4.7.3.2.12.3
4.2.9.2	Unsigned data type	ITS 4.7.3.2.12.3
4.2.9.3	Integer data type	ITS 4.7.3.2.12.3
4.2.9.4	Float data type	ITS 4.7.3.2.12.3
4.2.9.5	Logical data type	ITS 4.7.3.2.12.3
4.2.9.6	Array data	ITS 4.7.3.2.12.3
4.2.9.7	Multiple (overloading) types	ITS 4.7.3.2.12.3
4.2.9.8	Pointer access prohibited	EJ
4.2.10.1	Single IF ... ELSE structure	EJ
4.2.10.2	Single loop structure	EJ
4.2.10.3	Jump to label	EJ
4.2.10.4	Subprograms Implemented	EJ

4.2.10.4	Subprogram parameter conformance	EJ
4.2.10.4	Prohibit recursion	EJ
4.2.10.5	Suspend till next frame command	ITS 4.7.3.2.12.4
4.2.11.1	Assignment operator	EJ
4.2.11.2	Arithmetic operators	ITS 4.7.3.2.12.3
4.2.11.3	Bit manipulation	EJ
4.2.11.4	Logical combination	EJ
4.2.11.5	Relational operators	EJ
4.2.12	Maths functions	EJ
4.2.13.1	Defined operator precedence	EJ
4.2.13.2	Defined expression evaluation	EJ
4.2.13.3	Brackets to override operator precedence	EJ
4.2.13.4	Evaluation of mixed type expressions	EJ
4.2.13.5	Treat underflow as zero	EJ
4.2.13.6	Flag overflow	EJ
4.2.14	SAIL High Level Specification Document	EJ
4.2.15	Access to clock information	EJ
4.3.1	Up to 16 tasks	EJ
4.3.2	All tasks/clock period	ITS 4.7.3.2.12.4
4.3.3	Task over-run	EJ
4.3.4	Inter-task communication	ITS 4.7.3.2.12.3
4.3.5	Executive documentation	EJ
4.4.1	SAIL code < 65536 bytes	EJ
4.4.2	SAIL variables < 65536 bytes	EJ
4.4.3	Memory allocation	EJ
4.4.4	Memory protection	EJ
4.4.5	Spare task storage	EJ
4.5.1	SAIL provides memory management data	ITS 4.7.3.2.14
4.5.2	Starting and stopping tasks from ground	ITS 4.7.3.2.12.3
4.5.3	Stopping tasks from SAIL	ITS 4.7.3.2.12.3
~~4.5.4~~	~~Checksum procedure~~	~~ITS 3.10.2.5~~
~~4.5.4~~	~~ITS 4.7.3.2.13~~	
4.5.5	Complete tasks and data tables	EJ

6. List of Acronyms

AMPTE	Active Magnetospheric Particle Tracers Experiment
CCE	Charge Composition Explorer spacecraft
HPCE	Lockheed Hot Plasma Composition Explorer
CSS	Cooler Subsystem
EJ	Engineering Judgement
FSS	Flight software simulator
GGS	Global Geospace Science Mission
GSS	Gyro Subsystem
HIRDLS	High Resolution Dynamics Limb Sounder
HPCE	Lockheed Hot Plasma Composition Explorer instrument
ISAMS	Improved Stratospheric and Mesospheric Sounder
LMPARL	Lockheed Martin Palo Alto Research Laboratory
ICD	Interface Control Document
ICP	Instrument Control Parameter
IFC	In-Flight Calibration Subsystem
IPS	Instrument Processor Subsystem
IPU	Instrument Processor Unit

IST	Instrument Support Terminal
ITS	Instrument Technical Specification
I&T	Integration and Test
PCL	Program Control Language
PSS	Power Subsystem
SAIL	Science Algorithm Implementation Language
SPU	Signal Processing Unit
SRD	Science Requirements Document
TSS	Telescope Subsystem
TIMAS	Toroidal Imaging Mass-Angle Spectrograph